分散データベース・システム設計の問題点

國井 利泰
東京大学理学部情報科学科

はじめに

私たちは、現在、分散型データベースシステムとして、ミニコンをベースとしたものを考えている。今日は、この分散型データベースシステムをソフトウェア工学上のテーマとして話しをしようと思う。特に、実際の設計のためのソフトウェア工学的アプローチに重点を置き、分散型データベースシステムの設計という面から話したい。

分散データベース・システム（Distributed Database Systems, DDSs）は、云うまでもなく分散処理システム（Distributed Processing Systems, DPSs）の一種である。DDSs は DPSs に共通の目的を加えて、データ共有化という独自の主たる目的にしている。

データベースの場合、分散という時に、一般には単に物理的データ分散を指すのみならず、データ表現ならびにデータ検索要求などの論理的データ分散をも併せて意味する。したがって、データ共有化も、物理的共有化と論理的共有化の両者を達成する必要がある。

時にデータ表現の共通化、特にデータ翻訳が論理的データ共有化の問題として議論されることがあるが、これは実は全問題のほんの一面である。以下、分散データベース・システムの全体像を明らかにすべく議論を展開する。

現在の分散型データベースシステムの分類

現在の分散型データベースシステムを分類してみると、第1の分類は、その視点を応用（Applications）、機械（Machines）、データベース（Databases）及びシステム制御（System Controller）の4つの論理的な要素からみた場合の分類である。

表1　応用、機械、データベース及びシステム制御面からみた場合の分散型データベースシステムの分類

視　点	分　類	システム例
応用 (Application)	汎用(General purpose)	ARPANET-Datacomputer
	専用(Special purpose) (Dedicated)	PARS ESIS
機械(Machine)	異機種(Heterogeneous)	ARPANET-Datacomputer
	同機種(Homogeneous)	HP-3000 MININET (Univ. Waterloo)
データベース (Database)	多様(Diversified)	ARPANET-Datacomputer MIT CONIT
	一様(Uniform)	ほとんどのデータベースシステム
システム制御 (System Control)	分散制御(Decentralized control)	CCA SDD-1
	集中制御(Centralized control)	SRI LADDER-FAM

ミニコンベースの分散型データベースシステムは、現在、まだSDD-1とHP-3000がはしりで、アメリカにおいてもこれからという時代である。このようなシステムを開発しようとすると、どうしても通信と計算機の両面の技術を持っていなければならないために、開発する企業は限定されるであろう。

第2の分類は、設計目的からの分類である。これは2種類に分類することができる。

1つは、データベースの共有（Database sharing）を目的とするコンポジショナルな設計タイプであり、今1つは、スループット／応答性、応用に関する機能性、信頼性／保守性等の適応性（Adaptability）を目的とするディコンポジショナルな設計タイプになる。この分類を表2に示すとこのようになる。

表2　分散型データベースシステムの設計タイプによる分類

設計タイプ	設計目的	システム例
Compositional	データベース共有 (Database Sharing)	MIT CONIT ARPANET-Datacomputer UCLA-IBM "DBMSs Communication" Project
Decompositional	適応性(Adaptability)	
	(Throughput/ Responsiveness) ・スループット／応答性	MIT INFOPLEX Multiprocessing Backend Database Computer (1)
	(Functionality (Application Orientation)) ・応用に関する機能性	MIT INFOPLEX
	(Reliability/ Survivability) ・信頼性／保守性	CCA SDD-1

注(1)　E. I. Lowenthal による提案

コンポジショナルな設計は、単に、サブシステム間でデータベースを共有しようとするものである。

ディコンポジショナルな設計については、簡単な例を考えてみると、大型計算機を用いて1つのシステムにするかわりに、ミニコン2～3台でデータベースを実現させると、信頼性は増大する。なぜなら、1台のミニコンがダウンしても他のミニコンは生き残っているから、処理速度が遅くなるだけでシステム全体としてはダウンしない。この点から信頼性は、増大すると考えられる。

最近アメリカでは、大型計算機メーカーに対抗するために、小型計算機あるいはミニコンを何台か接続して、非常に信頼性が高く、しかも応答時間が早くなるようなシステム指向が強くなってきている。IBMが最近やっと小型計算機にカを入れだしてきたのも以上のような理由からである。

それでは、コンポジショナルな設計タイプとディコンポジショナルな設計タイプとはどのようなものであろうか。概念図を示すと図1のようになる。

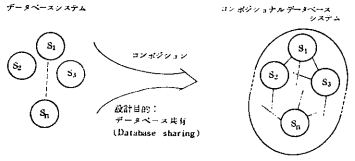


図 1.1. コンポジショナルデータベースシステム

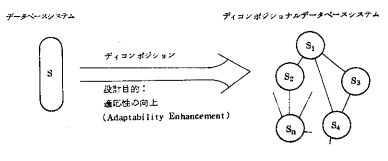


図 1.2. ディコンポジショナルデータベースシステム

ディコンポジショナルな設計アプローチと設計基準

分散型データベースシステムを考える場合、物理的に分散化しているということのほかに、論理的な分散化ということに注意しなければならない。物理的分散（Physical distribution）は、単に地理的に異なった点にデータベースが分散しているということのみで、なんら設計上の障害にはならない。次に論理的な分散（Logical distribution）についてであるが、ここで私が言う論理的という意味は、例示すれば次のようなことである。

① データ表現は何通りあるか。

② 質問言語は何通りあるか。

同じデータでも、例えば応用別に違った質問言語でユーザが問いてくると、応用別の違った論理単位に分かれてしまい、論理的な分離が発生する。多分このことが、ソフトウェア工学上の中心的な問題であると思う。

③ 量的な問題。

- 1表現当たりの情報量。
- 1質問当たりの情報量。
- 1質問言語当たり何通りの単語があるか。（現在、1言語当たり最低でも12個以上はある。）

以上のような点に重点がおかれなければ、いかに物理的に接続されていても分散したサブシステム間の実際に意味を持ったコミュニケーションは不可能となり、分散型データベースシステムとは成り得ない。これらの点から、分散型データベースシステムを統合したり、分解したりする場合の基準として次の2点をあげることができる。

① 統合する場合の基準としては、データの翻訳量を極力少なくするということである。

このためには、共通の質問言語を作ることと、データ表現を共通化することの2点を重点的に行なっていくのが好ましいと思う。IBMのプロジェクトでは、伝統的にサブシステム間でデータ翻訳を続けていく方法を基に研究・開発が行なわれているが、これでは翻訳量の増大につながり、好ましくない。

② 分解する場合の基準としては、Maximum Locality の原則がある。

これについては、"最もよく使用される場所に、よく使用する論理単位を置く"ということであり、言い換えれば、サブシステム間の不必要なトラフィックをできるだけ少なくするということにある。

物理的分散と論理的分散の概念を図2に示そう。

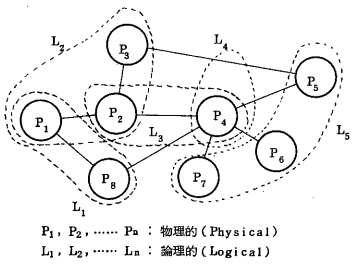


図2　物理的分散と論理的分散

次に、今までのことを踏まえて、分散型データベースシステムを作成するためのソフトウェア工学的アプローチについて話していこうと思う。

分散型データベースシステム支援ソフトウェア・ツール群の役割

ソフトウェア工学的アプローチとは何か。これは、個々のシステム（例えば、コンパイラ、OS等）を直接作成するのではなく、まず個々を作成するためのツールから作っていくことだと考える。つまり、分散型データベースシステムを直接作成するのではなく、システムを作るにはまずどんなソフトウェアサポートツールが必要かを考え、それを作ってから、それを使って目的のシステムを作成していくことである。

ツールを作成し、そのツールで個々のシステムを自動生産していくと次のようなメリットがある。

① エラー発生率は少なくなる。
② エラー率は統計的に管理でき、品質管理ができる。
③ 個々のシステム作成コストが安くなる。

どのようなツールを用いたら良いか。これについては、一般論はいろいろあるが、具体的なものについてはあまり言われていない。そこで、分散型データベースシステムに関してのツールについて、これから説明していきたいと思う。

まず、私たちが考えているデータベースシステム（以下DSと略す）から話していこう。

(1) 私たちの考えるDS

私たちが目標とする分散型データベースシステムは、高水準なデータベースシステムで次のようなものと考えている。

- アプリケーション的側面……どのようなアプリケーションにも対応できるもの。
- データ管理的側面……データの構築をユーザが考える必要のないもの。すなわち、データの論理的な記述を入力すると計算機が自動的にデータ構築をしてしまうもの。
- 機械的側面……どのような機械がいつ接続されても対処できるようなもの。

したがって、このような特徴をもつDSを作成するためのツールを考える必要がある。

では、現在あるDSは、どのような水準にあるのかについて述べてみたいと思う。

(2) 現在あるDSの水準別の分類

まず現在あるDSを水準別に分類すると表3のように、高水準なものと、低水準なものに分類することができる。

低水準なものの典型的な例としてIMS（IBM）を考えてみると、このシステムではデータ接近経路など、個々のユーザごとのデータ管理にまでメーカのソフトウェア技術者が手直しする必要があり、メンテナンス

表3　　DSの水準別分類

レベル	例	データ管理
高水準	ADABAS	表管理のみ
	SYSTEM 2000	表管理とアクセスパスの指定による管理の両方の機能をもつ
低水準	IMS IDS IDMS TOTAL	アクセスパスの指定による管理のみ

工数が全体の75%以上になっている。一般にメンテナンス工数は、ソフトウェアライフサイクルのうちの約75%位である。これは一期ごとに約20%強の人員がメンテナンスにとられるとして、6期経過したという、ごく普通の場合を想定している。

このことをソフトウェア工学的に考えると、この問題を解決してメンテナンスの必要がないシステムを作成する方向を見い出していかなければならない。したがって、このためのソフトウェア作成ツールを作ることが、ソフトウェア工学になろう。

以上のことをDSに適用してみると、データベースのメンテナンスはデータの再構造化であり、このデータの再構造化を防ぐためのツールを作成する必要があるということになる。

したがって、データの再構造化を防ぐことが私たちの当面の目標である。

ここで具体的にデータの構造化の例をあげ、高水準と低水準のDSで比較してみよう。

（例）COBOLでデータ構造を記述すると、仮に10,000ステップ要したとすると

- 高水準では、400ステップ要する。
- 低水準（IMSの場合）では、2,000ステップ要する。

このように高水準の場合には、ステップ数が非常に少なくなる。このことは、人間の介入によるエラーの減少化にもつながる。さらに、後に述べるように高水準データベースシステムでは、データを記述するプログラムの書換え等人手によるデータの再構造化の必要がないために、プログラム（アプリケーション）を変更する必要もない。メンテナンスコストを減少するためには、これからの分散型データベースシステムは高水準なもの以外は考えられない。

このことについてE.F.Codd（IBM）は、次のようにデータの独立性の達成の必要性を提言した。

"データが他のもの（アプリケーション、機械）と独立でないデータベースシステムは存在すべきではない。"

しかし、私たちの考える分散型データベースシステムにおいては、このことだけでは不十分である。後で、分散型データベースシステムは本来どうあるべきかについて述べていきたいと思うが、このことが主に、分散型データベースシステムにおけるソフトウェア工学的側面になると考える。

これから私たちの考える分散型データベースシステムの形態を明確にしたい。

分散型データベースシステムの形態—開放型システム・アーキテクチャ、並びにその仮想データベース・システムとしての実現—

ユーザから分散したデータベースをみると図3のように、応用上必要な1つの仮想的なデータベースであり、それは論理的なものである。この点からみると、ユーザが使用したいデータベースは物理的にはどこにあってもかまわないわけである。

分散型データベースシステムの特徴をあげると、次の3つがある。

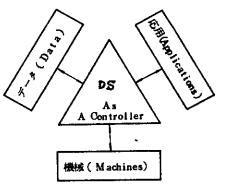


図3　　DS環境

① データは進化する……データ（D）
② アプリケーションは変化する……アプリケーション（A）
③ 機械は進化する……機械（M）

このように、分散型データベースシステムは3つの要素が相互に進化するという新しい特徴を含んでいる。このような分散型データベースシステムにはそれが一つの統一のとれたシステムとして存在するために、本来分散システム全体の大局オペレーティングシステム（GOS）がなければならない。次に制御系、アプリケーション、データ、機械の各サブシステムに必要な条件について明らかにしていく。

(1) 大局制御系（GOS）

制御系には、次の3つの機能が必要である。

① 3つの論理的なサブシステム（A，M，D）の間のマッピング機能。
マッピングするためには、インターフェースジェネレータが必要になる。これにより、データ（D）も機械（M）もアプリケーション（A）も仮想化される。つまり、D、M、Aの3つのサブシステムがダイナミックに変化（進化）するため、各々を仮想化（バーチャライズ）するジェネレータがないと、制御系自身が使えなくなってしまう。具体的な例をあげると、
- 機械に関しての仮想化は、VM（Virtual Machine）としてかなり進んでいる。IBMのVM、ハネウェルのVMなどがある。
- データに関しては、例えば、ADABASで仮想されている。
- アプリケーションに関しての仮想化は、良いアプリケーション記述言語がないためにまだ満足がいくようには行なわれていない。

② ダイナミックなマッピングを制御するスケジューラ機能。
③ マッピングを最適化する機能。

以上の機能を満足するような大局制御系が存在することが、1つの分散型データベースシステムのシステムたり得る所以である。

(2) アプリケーション

アプリケーションについては、ユーザ要求の明確化を行なうことにある。アプリケーションの記述を手続き的プログラムで表現すると、その表現は、処理もデータもごっちゃに含むのでこれは良い方法ではない。したがって、前述のマッピングを可能とするためには非手続き的にアプリケーションを明確にしていく方法を考えていけば良いと言える。

現在までに提言されている方法のうち、非手続き向きに近いものでアプリケーションを明確にするための方法には、次の4つのものがある。

① 階層的グラフ形式（Hierarchical graph formalism）
（例）SADT of SofTech
② 並列グラフ形式とその変形（Bipartite graph formalism and its modification）
（例）RSL of TRW（ペトリネット）
③ 階層的モジュール設計技術（Hierachical modular Design Technigues）
④ 非手続的質問言語（Query Languages）
（例）CoddのALPHA言語等

(3) データ

データは、データ値そのものとデータの構造（関係）の2つから成り立っている。また、データの格納方法にはデータそのものを格納する方法とデータを算出するアルゴリズムを格納していく方法とがある。前者は、メモリ容量の無駄につ

ながり、後者は、メモリ容量の無駄を省くことができるが、処理時間がかかる。

データの正しさを保証するためには、データをコントロールするためのルール（規則）を作成し、それをデータをデータベースに入れる際に利用すればよいと思われる。例えば、データベースをユーザに提供する場合には、そのルールを作成しておき、このルールに従わなければデータの変更等はできないようにしておくべきだと考える。現在それは、次の3通りの仕方でインプリメントされている：

① 手続き的にプログラムとして：現在最も多く使われる方法
COBOL などプログラム言語により、手続き的に条件文、論理文等により書く。

② データ・モデルにより：次第に使われる頻度がふえている。
C. Bachman's diagrams, P. Chen's E-R diagrams 等のデータ・モデルを用いて書く。

③ 非手続き的に述語論理を用いて：研究段階
M. Hammer 等のやっているように、述語論理を基本とした言語により、非手続き的表現として書く。

(4) 機械

ユーザから機械をみた場合には、アプリケーション上の操作に近いようにみえる。例えば、事務用マシンとか技術計算用マシンとかいうようにみえる。したがって、分散型データベースシステムを作成する場合には、ユーザ1は、アプリケーションに関係した部分以外は隠した方がよいと言える。

分散型データベースシステムの設計基準・システム設計法

以上のようなことから、私たちの考える分散型データベースシステムの設計条件をまとめると次の3つの条件を満足させることにある。

(1) 3つの論理サブシステムのそれぞれの変化が、それぞれのサブシステム内の変化しない部分に波及しないこと。

(2) 3つの論理サブシステムの変化が相互に波及しないこと。つまり各サブシステムがお互いに独立である。

(3) 制御系は、3つの論理サブシステムの変化に対して不変であること。

このことは分散型データベースシステムの設計にあたってそのようなシステムが存在し得るための存在公理とも言えることである。

次に、この2つの条件を満足させるための方法について述べたいと思う。

私たちが考えるDSを設計するためには、前述の3つの条件を満足させるように作成する。

(1) 第1の条件（各サブシステムの自己独立）を満足するための方法。

各サブシステム構成の出発点となり得る論理的な要素と、その構成規則とを仕様化し、同時に自動化した構成系を用意することにより第一の条件を満たすことができる。一般に構成規則は再帰的になる。与えられた環境において再帰的なものを階層構造などの非再帰的なものに展開して実行性能向上を計る各種の方法が提案されている。

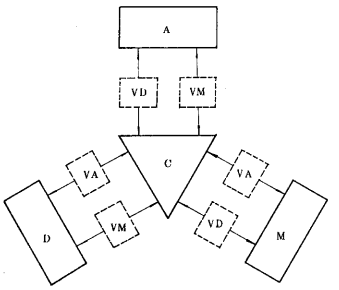
(2) 第2の条件（各サブシステムの相互独立）を満足するための方法。

各論理サブシステムから相互に他がどのようにみえるかという観点を提供するバーチャライザを作成すればよいわけである。これにより、各論理サブシステム

の仮想化ができる。このようなバーチャライザにより、各論理サブシステムはたいに独立になる。しかし、これだけでは制御系が論理サブシステムの変化により変ってしまう。そこで、第3の条件が必要になる。

(3) 第3の条件（制御系の不変化）を満足するための方法。

制御系の中にバーチャライザのジェネレータを含めて、サブシステムの変化による制御系の変化分をバーチャライザの更新部分として制御系の外に出してしまうことにより、制御系を3つの論理サブシステムの変化に対して不変にすることが可能になる。



A：応　用（Applications）　VA：仮想応用（Virtual Applications）
D：データ（Data）　VD：仮想データ（Virtual Data）
M：機　械（Machines）　VM：仮想機械（Virtual Machines）
C：制御系（Controller）

図4　分散型データベースシステムのVirtual View

バーチャライザ・ジェネレータの実現例

次にバーチャライザ・ジェネレータの実現方法について述べよう。

データのバーチャライザ・ジェネレータの実現方法については、CONIT（MIT）の論文に多少これに近いことが記載されているので参考にしてほしい。

ここでは、応用のバーチャライザの一例としてADABASとSYSTEM 2000を紹介する。

(a) ADABASの例

ADABASではバーチャライザをアソシエータと呼んでいる。ADABASのデータ管理は表管理のみである。システムがジェネレータにより応用からデータへの接近経路（アクセスパス）をアソシエータとして生成してくれるので、その指定の必要はない。したがって、このアーキテクチャにより、難問が単純化されている。

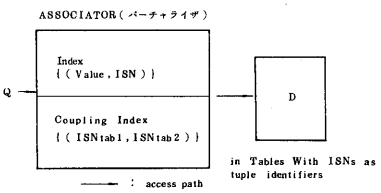


図5　ADABASにおけるバーチャライザの構造

このことをソフトウェア工学的にみると、物事の単純化、余計なことはしない、何をするかを明確にして自動化するという点につながる。

(b) SYSTEM 2000の例

SYSTEM 2000のデータ管理は、表管理の他にアクセスパスの指定ができるようになっている。しかし、アクセスパスの指定ができるために構造は複雑になっている。アクセスの方法が固定化している場合には、アクセスを高速化するという意味で、アクセスパスの指定は有効であるが、今後はアクセス最適化モジュールの

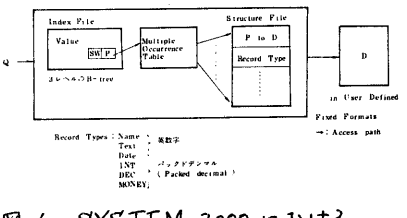


図6　SYSTEM 2000におけるバーチャライザの構造

開発技術の進歩により次第にその必要性が減少することも予想され、これは分散型データベースシステムの過渡期にのみいえることだと思う。

おわりに

私たちの考えているソフトウェア工学とは、今まで述べてきたような極めて実際的な考え方であり、今後一層必要性が増大し、また実現性も増大すると思う。

要点をまとめると、次のようになる。

① どのようにしたら簡単になるかを考えること。
② 要求は何かを明確にすること。
③ 自動化できるものは自動化すること。

REFERENCES

/BOOTH 76/ G.M.Booth, "Distributed Data Bases-Their Structure and Use," Infotech State of the Art Report, Distributed Systems, pp.201-213, 1976.

/CODD 70/ E.F.Codd, "A Relational Model of Data for Large Shared Data Bank," Comm. ACM, 13, pp.377-387,1970.

/DATE 75/ C.J.Date, "An Introduction to Database Systems," Addison-Wesley, Reading, Mass., 1975.

/DAVIS 77/ R.Davis, B.Buchanan, and E.Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultant Program," Airtificial Intelligence, 8, pp.15-45, 1977.

/EPSTEIN 76/ H.A.Epstein, BALLOT project, SCIP, Stanford University, Private Communications, 1976.

/HEART 70/ F.E.Heart, R.E.Kahn, S.M.Ornstein, W.R.Crowther, and D.C.Walden, "The Interface Message Processor for the ARPA Computer Network," Poc. AFIPS Spring Joint Comp. Conf. pp.551-567, 1970.

/HOUSEL 74/ B.C.Housel, V.Y.Lum, and N.C.Shu, "Architecture to An Interactive Migration System (AIMS), ACM SIGMOD Workshop on Data Description, Access and Control, pp.157-169, 1974.

/HP 75/ HP 3000 Data Center, Hewlett-Packard Database Systems Seminar Material, 1975.

/KNIGHT 72/ J.R.Knight, "Case Study: Airlines Reservations Systems," Proc. IEEE, pp.1423-1481, 1972.

/KUNII 77a/ T.L.Kunii and H.S.Kunii, "Database Design," Proc. 10th Hawii Int. Conf. System Sciences, pp.200-203, 1977.

/KUNII 77b/ T.L.Kunii, "Data Sharing Models in Distributed Database Systems," Rep. Spec. Res. Proj. on Formation Process of Inf. Sys. and Organization of Sc. Inf., Sponsored by Min. Education, Sc. and Culture, Gov. Japan, 1977.

/KUNII 77c/ T.L.Kunii, "Database System Design Criteria, Part 1, Specification of Basic Design Requirements for Application Independence," Inf. Proc. Soc. Japan, Software Eng. Working Group Tech. Memo, No.1, Part 2, pp.1-7, 1977.

/LOWENTHAL 76/ E.I.Lowenthal, "The Backend (Data Base) Computer - Part I, II," Data Base Management, 24-01-04,pp.1-12; 24-01-05, pp.1-16, Aurerbach Pub., 1976.

/MADNICK 75/ S.E.Madnick, INFOPLEX-Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex," Proc. AFIPS Nat. Comp. Conf., pp.581-586, 1975.

/MANNING 75/ E.Manning and R.W.Peebles, "A Distributed Operating System Nucleus for Transaction Processing on Distributed Data Bases," Proc. 4th Texas Conf. Computing Systems, pp.5A-2.1 - 5A-2.5, IEEE Comp. Soc., 1975.

/MARCUS 75a/ R.S.Marcus, "Networking Information Retrieval Systems Using Computer Interfaces," ASIS Proc., 12, pp.77-78,1975.

/MARCUS 75b/ R.S.Marcus, "A Translating Computer Interface for a Network of Heterogeneous Interactive Information Retrieval Systems," ACM SIGPLAN Notices, 10, pp.2-12,1975.

/MARILL 75/ T.Marill and D.Stern, "The Datacomputer - A Network Data Utility," Proc. AFIPS Nat. Comp. Conf., pp.389-395, 1975.

/MILLS 76/ H.D.Mills, "Software Development," Suppl. Proc. 2nd Int. Conf. Software Eng., pp.79-86,1976.

/MORRIS 77/ P.Morris and D.Sagalowicz, "Managing Network Access to a Distributed Database," Proc. 2nd Annual Berkeley Workshop on Distributed Data Management and Computer Networks, pp.58-67, 1977; also available as SRI AIC Tech. Note 144, Menlo Park, Calif.

/NAHOURAII 76/ E.Nahouraii, L.O.Brooks, and A.F.Cardenas, "An Approach to Data Communication between Different Generalized Data Base Management System," System for Very Large Data Bases, Lockemann and Neuheld, eds., pp. 117-142, North-Holland, Amsterdam, 1976.

/PEEBLES 75/ R.Peebles and E.Manning, "A Computer Architecture for Large Distributed Data Bases," Proc. 1st Int. Conf. Very Large Data Bases, pp.405-427, 1975.

/ROTHNIE 77/ J.B.Rothnie and N.Goodman, "An Overview of the Preliminary Design of SDD-1: A System for Distributed Databases," Proc. 2nd Annual Berkeley Workshop on Distributed Data Management and Computer Networks, pp. 39-57, 1977. Proc. avail. from NTIS, Springfield, Virginia.

/SACERDOTI 77/ E.D.Sacerdoti, "Language Access to Distributed Data with Error Recovery," SRI AIC Tech. Note 140, Menlo Park Calif., 1977.

/SHATZ 73/ V.L.Shatz, "Computer Networks for Retail Stores," Computer, pp.21-25, 1973.

/SHU 77/ N.C.Shu, B.C.Housel, R.W.Taylor, S.P. Ghosh, and V.Y.Lum, "EXPRESS: A Data Extraction, Processing, and REStructuring System," ACM Trans. Database Systems, 2, pp.134-174, 1977.

/SWARTWOUT 77/ D.E.Swartwout, M.E.Deppe and J.P. Fry, "Operational Software for Restructuring Network Databases," Proc. AFIPS Nat. Comp. Conf., pp.499-508, 1977.

/THOMAS 75/ R.H.Thomas, "A Solution to the Update Problem for Multiple Copy Databases which Uses Distributed Control," BBN Report No. 3340, 1975.